tools/aapt/Resource.cpp

parsePackage

解析AndroidManifest.xml

将AndroidManifest.xml进行flatten

## AaptAssets类

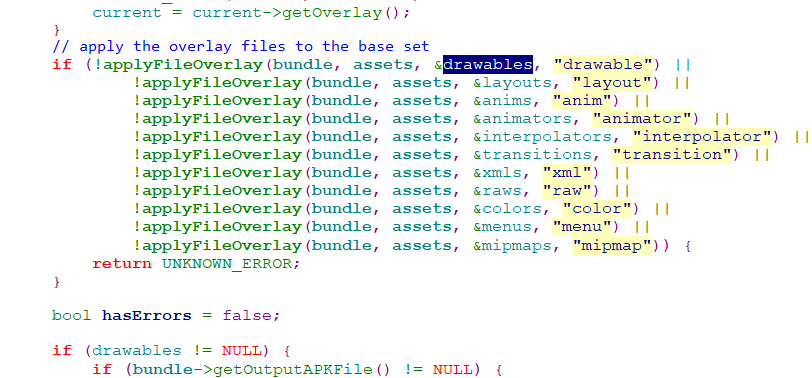
存储文件，不包含values

AssetManager mIncludedAssets，包含通过-i指定的已编译好的包，如android\sdk\platforms\android-27\android.jar

Vector<sp<AaptDir> > mResDirs，收集完成文件之后的资源目录，resType-》AaptDir，resType记录在AaptDir的mLeaf字段中

sp<AaptAssets> mOverlay 制定了多个res目录，第一个之后的都是overlay

applyFileOverlay用overlay的覆盖base中的，需要AaptGroupEntry一样，即配置信息一样。如果base中不错在，直接添加



**makeFileResources**作用，遍历ResourceTypeSet，将收集到的resource信息分别存入AaptAsset和ResourceTable

存储values和文件路径信息，对应resources.arsc

**compileResourceFile**编译values目录下的资源

调用parseXMLResource，然后

## AaptFile

getSourceFile()文件路径

## Package类

sp<AaptFile> mTypeStringsData;

sp<AaptFile> mKeyStringsData;

ResStringPool mTypeStrings;

ResStringPool mKeyStrings;

DefaultKeyedVector<String16, uint32\_t> mTypeStringsMapping; 类型字符串和typeId(在ResStringPool中的索引)的映射

DefaultKeyedVector<String16, uint32\_t> mKeyStringsMapping; 键值字符串和entryId(在ResStringPool中的索引)的映射

typeString

keyString?

valueString?

## StringPool类

**字符串vector，同一字符串只存在一个，entry 中的indices为该entry在mEntryArray中的index**

**Vector<entry> mEntries;**

**存储字符串在mEntries中的索引，如果同一字符串add了多次，在mEntryArray中就会存在多个元素，但是值相同，对应与mEntries中的同一个值**

**Vector<size\_t> mEntryArray;**

**元素与mEntryArray对应，表示对应的entry的style，一个entry会有多个style**

**Vector<entry\_style> mEntryStyleArray;**

**字符串，ssize\_t位该字符串对应于mEntryArray的索引**

**DefaultKeyedVector<String16, ssize\_t> mValues;**

**add函数返回的是字符串对应于mEntryArray的索引，mEntryArray[pos]**

**StringPool::writeStringBlock**

**header->header.type = htods(RES\_STRING\_POOL\_TYPE);**

**header->header.headerSize = htods(sizeof(\*header));**

**header->header.size = htodl(pool->getSize());**

**header->stringCount = htodl(ENTRIES);**

**header->styleCount = htodl(STYLES);**

**if (mUTF8) {**

**header->flags |= htodl(ResStringPool\_header::UTF8\_FLAG);**

**}**

**header->stringsStart = htodl(preSize);**

**header->stylesStart = htodl(STYLES > 0 ? (preSize+strPos) : 0);**

**// Write string index array**

**// Write style index array**

**// Write strings**

**// Write styles**

**attr.nameResId**

**attr.namePoolIdx**

**resids**

**XMLNode::flatten**

**slurpFromArgs解析参数，收集文件**

****

## buildResources

**parsePackage**

解析AndroidManifest.xml

获取包名，填入AaptAsset的mPackage中

**ResourceTable table(bundle, String16(assets->getPackage()), packageType);**

构造ResourceTable，设置packageId

App和AppFeature的packageId为0x7f

System的packageId为0x01

SharedLibrary的packageId为0x00

**ResourceTable::addIncludedResources**

添加依赖资源包，比如系统资源包android.jar，里面包含系统资源的定义

**collect\_files(assets,** **resources)**

将所有的资源文件保存到resources，然后保存到

sp<ResourceTypeSet> drawables;

sp<ResourceTypeSet> layouts;

sp<ResourceTypeSet> anims;

sp<ResourceTypeSet> animators;

sp<ResourceTypeSet> interpolators;

sp<ResourceTypeSet> transitions;

sp<ResourceTypeSet> xmls;

sp<ResourceTypeSet> raws;

sp<ResourceTypeSet> colors;

sp<ResourceTypeSet> menus;

sp<ResourceTypeSet> mipmaps;

**applyFileOverlay**

**preProcessImages**

处理.9.png文件

**makeFileResources**

将资源加入AaptAssets和ResourceTable

**compileResourceFile**

编译res下的values资源

生成ResourceTable下的Package，Type，Entry，Bag，Item

**ResourceTable::assignResourceIds**

按资源出现的顺序为资源分配id(结合public中指定的值)

**compileXmlFile**

**postProcessImage**

**ResourceTable::addSymbols**

**生成R.java**

**ResourceTable::flatten**

**生成resources.arsc**

**ResTable::stringToValue**